Ukrainian Catholic University

Faculty of Applied Sciences

Data Science Master Programme

Optimization of the Transformer's attention

Linear Algebra final project report

Authors:
Anton Brazhnyi
Olexandr Korniienko
Andrii Ruda

28 February 2023

Abstract

Attention is a critical component of transformer models, but its computational complexity limits its scalability. This project investigates linear algebra-based approaches to optimize the attention mechanism's performance, including low-rank approximations and kernels/factorization methods. We provide a detailed analysis of the computational complexity and compare their empirical performance on an image classification task.

1 Introduction

The Transformer architecture has become a cornerstone in modern deep learning and has led to numerous breakthroughs in natural language processing, computer vision, and other areas. The Transformer's attention mechanism has been shown to be a crucial component in its success. However, computing the attention scores for all pairs of positions in an input sequence results in quadratic time and memory complexity with respect to the sequence length, making the mechanism slow and memory-intensive for long input sequences.

In this project, we explore various methods for optimizing the attention mechanism in the Transformer architecture. We focus on four methods:

1. Linear Attention
2. Linformer Attention
3. Random Feature Attention
4. Nystrom Attention

Each method uses linear algebra techniques to optimize the attention mechanism and reduce the computational and memory costs of the Transformer architecture. We will provide an overview of each method and compare their computational complexities and performance on an image classification task.

2 Quick methods overview

2.1 Softmax Attention

Attention calculates a weighted average of the feature representation with weights proportional to the similarity score between pairs of representations. Let's consider input $X^{n\times d}$ and $W_q \in R^{d\times d_q}$, $W_k \in R^{d\times d_k}$, $W_v \in R^{d\times d_v}$ referred as query, key, and value projectors for input sequence with n length. The projections (outputs) are then computed as $Q = XW_q$, $K = XW_k$, $V = XW_v$. The attention layer is defined as

$$\text{Attention}(Q, K, V) = \text{Softmax}\big(\frac{QK^T}{\sqrt{d_k}}\big)V$$

which requires $O(n^2 d)$ steps to compute and scales poorly with the large input size. An intuitive illustration of the Attention layer is presented in figure 1 below.

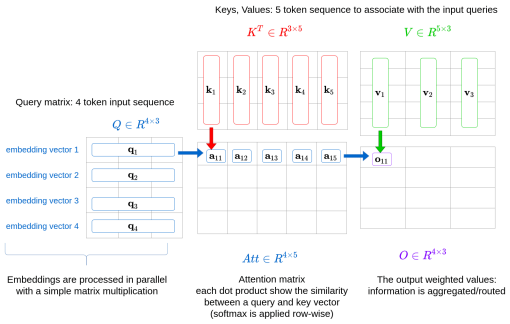


Figure 1: Illustration of self-attention mechanism [1].

However, the scaled dot-product attention isn't simply applied to the queries, keys and values. Instead of this, single attention components with: queries, keys and values are split into h heads, and the scaled dot-product attention is calculated over all heads in parallel.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$

$$\text{head}^i = \text{Attention}(QW_q^i, KW_k^i, VW_v^i)$$

The standard attention mechanism used in the Transformer architecture computes attention scores for all pairs of positions in the input sequence, resulting in quadratic time and memory complexity.

2.2 Linear Attention

Softmax dot-product attention has wide applications, but as we described in the previous section, its memory and computational costs grow quadratically with the input size. Such growth prohibits its application on high-resolution inputs. For this purpose, the efficient attention mechanism equivalent to the softmax attention is used. The Linear Attention mechanism allows for reducing memory and computational costs to remedy this drawback. At the same time, empirical evaluations [2] demonstrated the effectiveness of its advantages.

Linear attention proposes the individual feature vectors $X \in R^{n\times d}$ still pass through three linear layers to form the queries Q keys K, and values V. However, instead of interpreting the keys as n feature vectors in R^{d_k}, the module regards them as d_k single-channel feature maps. Linear attention uses each feature map as an overall weighting position and aggregates the value features through weighted summation to form a global context vector [3].

The equation describes linear attention presented below:

$$\text{Linear Attention}(Q, K, V) = \text{Softmax}_{row}\big(Q\big)\big(\text{Softmax}_{col}\big(K\big)^T V\big)$$

To prove the equivalence of the considered method to dot-product attention, let's replace Softmax normalizing function to $1/\sqrt{d}$, so the original formula takes a form:

$$\text{Attention}(Q, K, V) = \big(\frac{QK^T}{d_k}\big)V$$

Linear attention can be expressed in the following way:

$$\text{Linear Attention}(Q,K,V) = \big(\frac{Q}{\sqrt{(}d_k)}\big)\big(\frac{K^T}{\sqrt{(}d_k)}V\big) = \frac{1}{d_k}Q\big(K^TV\big) =$$

$$= \frac{1}{d_k}\big(QK^T\big)V = \text{Attention}(Q,K,V)$$

The main advantages of the algorithms are:

- improved computational complexity from $O(d * n^2)$ to $O(d^2 * n)$.
- increase memory efficiency from $O(n^2)$ to $O(dn + d^2)$ respectively.

2.3 Linformer Attention

Linformer Attention [4] uses low-rank approximations of the attention mechanism to reduce computational and memory costs by compressing the key and value vectors using linear projections.

Sinong at. al [4] introduced the Linformer Attention approach to improve memory and time complexity dot-product attention. Theoretical and empirical proof that the attention dot product matrix can be approximated by low-rank matrix with using the singular value decomposition (SVD) of the key and value matrices. Authors introduced a linear projection layer instead of using SVD algorithm to avoid calculating the decomposition for each attention matrix. For simplicity, the trainable projection layer parameters can be shared between attention heads and/or between layers.

The Linformer attention can be expressed as:

$$\text{Linformer Attention}(Q,K,V) = \text{Softmax}\big(\frac{Q\big(EK\big)^T}{\sqrt{d_k}}\big)\big(FV\big)$$

where $E, F \in R^{k\times n}$ are projection matrices of key and value $K, V \in R^{n\times d}$ to low-rank $R^{k\times d}$ matrices respectively.

To prove such possibility above, the paper provides a theoretical analysis of the spectrum of self-attention matrices and defines two theorems.

Theorem 1. For any Q, K, V there exists low-rank matrix $\tilde{P}$ such that

$$Pr(||\tilde{P}w^T - Pw^T|| < \epsilon||PW^T||) > 1 - o(1)$$

and

$$rank(\tilde{P}) = \theta(log(n))$$

where

$$P = \text{Softmax}\big(\frac{QK^T}{\sqrt{d_k}}\big)$$

Proof. Based on the definition of the context mapping matrix P, it can be expressed

$$P = \text{Softmax}\big(\frac{QK^T}{\sqrt{d_k}}\big) = \exp\left(A\right) \cdot D_A^{-1}$$

where $A = \frac{QK^T}{\sqrt{d_k}}$, D_A is $R^{n\times n}$ diagonal matrix.

Based on Johnson-Linderstrauss (JL) lemma [5] low-rank matrix $\tilde{P}$ can be constructed in the following way

$$\tilde{P} = exp(A)D_A^{-1}R^T R$$

where $R \in R^{k \times n}$ matrix with i.i.d. entries from $N(0, 1/k)$. JL lemma can be used to show that for any column vector $w \in R^n$ of matrix V, when $k = 5log(n)/(\epsilon^2 - \epsilon^3)$, we have:

$$Pr(||PR^T Rw^T - Pw^T|| < \epsilon||PW^T||) > 1 - o(1)$$

Theorem 2. For any $Q, K, V \in R_{n \times d}$ if $k = min\{\theta(9d \log d/\epsilon^2)\}$ there exists matrices $E, F \in R^{n \times k}$ such that for any row vector w of matrix $QK^T/\sqrt{d}$ we have:

$$Pr(||\text{Softmax}(wE^T)FV - \text{Softmax}(w)V|| < \epsilon||\text{Softmax}(w)|| \cdot ||V||) > 1 - o(1)$$

where

$$P = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})$$

The main advantages of the algorithm are:

- improved computational complexity from $O(d * n^2)$ (for dot-product attention) to $O(n)$ for Linformer attention
- increase memory efficiency from $O(n^2)$ to $O(n)$ respectively.

2.4 Random Feature Attention

Random Feature Attention (RFA) [6] is based on using random Fourier features to approximate a desired shift-invariant Gaussian kernel (softmax) [7] from dot-product attention. The method nonlinearly transforms a pair of vectors $\mathbf{x}$ and $\mathbf{y}$ using a random feature map ϕ. The inner product between $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$ can be approximated with the kernel evaluation on $\mathbf{x}$ and $\mathbf{y}$.

Theorem 3. [7] Let $\phi : R^d \to R^{2D}$ be a nonlinear transformation:

$$\phi(x) = \sqrt{1/D}\,[\sin(\mathbf{w}_1 \cdot \mathbf{x}), ..., \sin(\mathbf{w}_D \cdot \mathbf{x}), \cos(\mathbf{w}_1 \cdot \mathbf{x}), ..., \cos(\mathbf{w}_D \cdot \mathbf{x})]$$

. When d-dimensional random vector w_i are independently sampled from $\mathcal{N}(0, \sigma^2\mathbf{I}_d)$ and D - defines dimension of projection,

$$E\left[\phi(x) \cdot \phi(y)\right] = \exp\big(-||\mathbf{x} - \mathbf{y}||/2\sigma^2\big)$$

The variance of the estimation is inversely proportional to D (Appendix A.2 [8]).

The unbiased estimate of $\exp(< \cdot, \cdot >)$ can be estimated using theorem 3 and used for replacing $softmax$ layer in the dot-product attention:

$$\exp(x \cdot y/\sigma^2) = \exp(||x||^2/2\sigma^2 + ||y||^2/2\sigma^2)\exp(-||x - y||^2/2\sigma^2) \approx$$

$$\approx \exp(||x||^2/2\sigma^2 + ||y||^2/2\sigma^2)\phi(x) \cdot \phi(y)$$

Let's substitute softmax with random Fourier features in attention expression:

$$P = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}}) = \sum_i \frac{\exp(q \cdot k_i/\sigma^2)}{\sum_j \exp(q \cdot k_j/\sigma^2}) v_i^T \approx$$

$$\approx \sum_i \frac{\phi(q)^T \phi(k_i) v_i^T}{\sum j \phi(q) \cdot \phi(k_j)} = \frac{\phi(q)^T \sum_i \phi(k_i) \otimes v_i}{\phi(q) \cdot \sum_j \phi(k)}$$

where $\otimes$ denotes the outer product between vectors, σ^2 corresponds to the temperature.

The advantages of the proposed algorithm are:

RFA has a better memory efficiency than softmax attention with softmax normalization (linear vs. quadratic) i.e. $\phi(q)$, $\sum_i \phi(k_i) \otimes v_i$ and $\sum_i \phi(k_i)$ requires space of O(4D + 2Dd). From the opposite side, soft dot-product attention requires O(Nd).

2.5 Nystrom Attention

The Nystromformer architecture is designed to scale linearly with the input sequence length n. To achieve this, it employs the Nystrom method, redesigned for efficiency in approximating self-attention. Specifically, the Nystromformer algorithm selects landmark (or Nystrom) points to reconstruct the softmax matrix in self-attention, rather than computing the full $n \times n$ softmax matrix

According to the research [9], softmax matrix S could be written with the use of the Nystrom method as follows:

$$S = \text{softmax}(\frac{QK^T}{\sqrt{d_q}}) = \begin{bmatrix} A_S & B_S \\ F_S & C_S \end{bmatrix}$$

where $A_S \in R^{m \times m}$, $B_S \in R^{m \times (n-m)}$, $F_S \in R^{(n-m) \times m}$ and $C_S \in R^{(n-m) \times (n-m)}$.

We can approximate matrix $\hat{S}$ as following:

$$\hat{S} = \text{softmax}(\frac{Q\tilde{K}^T}{\sqrt{d_q}})(\text{softmax}(\frac{\tilde{Q}\tilde{K}^T}{\sqrt{d_q}})^{\dagger} \text{softmax}(\frac{\tilde{Q}K^T}{\sqrt{d_q}})$$

where $\tilde{K}$ and $\tilde{Q}$ are selected landmarks from K and Q, and † denotes Moore-Penrose pseudoinverse.

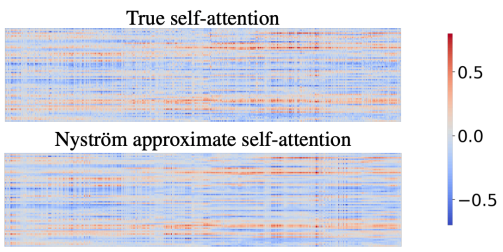


Figure 2: Illustration of Nystrom approximation vs pure self-attention. They are not identical but similar.

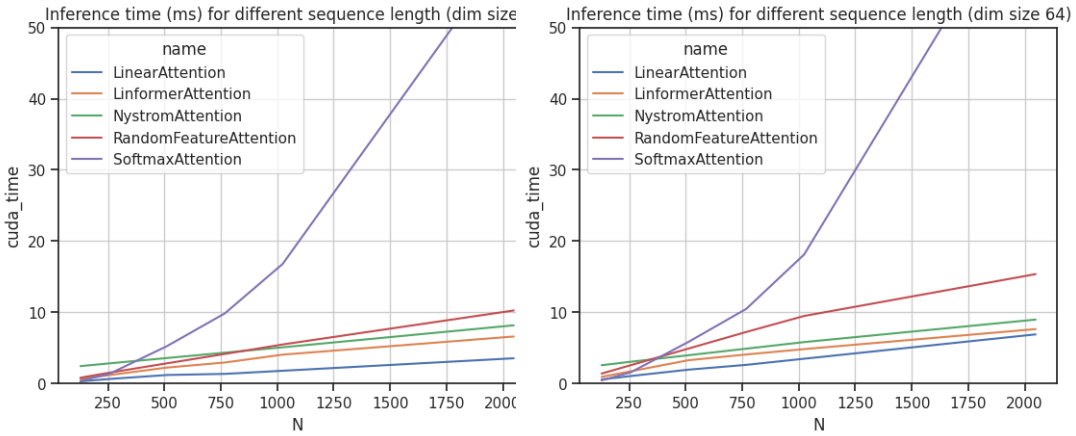
Method	Computational	Memory
Dot-product (softmax) attention	$O(d^2n + dn^2)$	$O(nd^2)$
Linear attention	$O(d^2n)$	$O(dn + d^2)$
Linformer Attention	$O(n)$	$O(n)$
Random Feature Attention	$O(nd)$	$O(4D + 2Dd)$
Nystrom Attention	$O(n)$	$O(n)$

Table 1: Methods complexity

3 Results

3.1 Computational complexity

This section analyses the efficiency advantage of attention approaches considered in this research over softmax attention in time and memory complexities. To measure computational complexity, we measure the inference time that it takes to compute values on NVIDIA GPU T4 for input data $X \in R^{batch \times head \times n \times d}$, where **batch** $= 32$, **heads** $= 4$, sequence length $\mathbf{n} \in \{128, 256, 512, 768, 1024, 2048\}$ and features dimension size $\mathbf{d} \in \{32, 64\}$. Typically sequence length is defined as image pixel size, and equals 1024 for image 32x32.



(a) N=32 (b) N=64

Figure 3: Computation time on GPU in ms.

3.2 Effect on accuracy

We report the validation accuracy of observed models on the CIFAR-10 [10] benchmark for the 10-class image classification task. We follow the standard evaluation protocols, train the model on the training set, report the results on the validation set, and compare them to our baseline - dot-product attention. All models are trained on Tesla A100 GPUs with 50 epochs (8k updates). The results of model accuracy depend on the iteration step and are presented in figure 4.

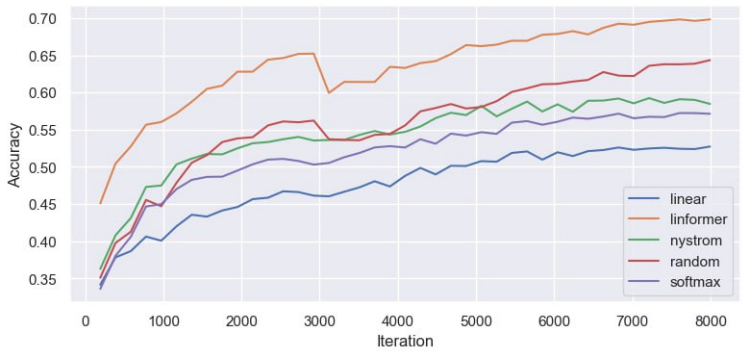


Figure 4: Computation time on GPU in ms.

3.3 Implementation details

Our models are trained with the cross-entropy objective. We use a batch size of 256, Adam optimizer with learning rate 5e-4, and cosine annealing scheduler, optimal parameters were estimated automatically by the Torch Lightning tool [1]. Input image size is 32x32, dot-product multi-attention embedding size 256 with 8 heads. We reduced embedding size by 8 times for Linformer, and selected a fixed number of landmarks for Nystrom attention equal to 64. Please find more details on the project github [2].

4 Conclusions

We investigated several methods of dot-product attention mechanism optimization in this approach using linear algebra matrix transformation techniques. It was empirically shown, that the linear attention approach has smaller computational complexity (execution speed) compared to observed methods. The Linformer attention mechanism showed the best classification accuracy compared to other observer models with similar hyper-parameters sets. Our code and setup are available at `https://github.com/fox-rudie/la-final-project-DS-at-UCU/tree/dev`.

References

[1] Adaloglou, N. (2021, March 25). Why multi-head self attention works: Math, intuitions and 10+1 hidden insights. AI Summer. Retrieved March 4, 2023, from https://theaisummer.com/self-attention/

[2] Katharopoulos, A., Vyas, A., Pappas, N., Fleuret, F. (2020). Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. ArXiv. https://doi.org/10.48550/arXiv.2006.16236

[3] Shen, Z., Zhang, M., Zhao, H., Yi, S., Li, H. (2018). Efficient Attention: Attention with Linear Complexities. ArXiv. https://doi.org/10.48550/arXiv.1812.01243

[1] `https://pytorch-lightning.readthedocs.io/en/stable/`
[2] `https://github.com/fox-rudie/la-final-project-DS-at-UCU/tree/dev`

[4] Wang, S., Li, B. Z., Khabsa, M., Fang, H., Ma, H. (2020). Linformer: Self-Attention with Linear Complexity. ArXiv. https://doi.org/10.48550/arXiv.2006.04768

[5] Ghojogh, B., Ghodsi, A., Karray, F., Crowley, M. (2021). Johnson-Lindenstrauss Lemma, Linear and Nonlinear Random Projections, Random Fourier Features, and Random Kitchen Sinks: Tutorial and Survey. ArXiv. https://doi.org/10.48550/arXiv.2108.04172

[6] Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N. A., Kong, L. (2021). Random Feature Attention. ArXiv. https://doi.org/10.48550/arXiv.2103.02143

[7] Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. In Proceedings of the 20th International Conference on Neural Information Processing Systems (NIPS'07). Curran Associates Inc., Red Hook, NY, USA, 1177–1184.

[8] Yu, F. X., Suresh, A. T., Choromanski, K., Kumar, S. (2016). Orthogonal Random Features. ArXiv. https://doi.org/10.48550/arXiv.1610.09072

[9] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li and Vikas Singh. Nyströmformer: A Nyström-Based Algorithm for Approximating Self-Attention, 2021; arXiv:2102.03902.

[10] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[11] Razavi, M. K.; Kerayechian, A.; Gachpazan, M.; and Shateyi, S. 2014. A new iterative method for finding approximate inverses of complex matrices. In Abstract and Applied Analysis.